PATENTS
130051-0078
MWS-056

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re The Application of: Peter Szpak et al.	
Serial No.: 10/759,346	Examiner: Ke, Peng
Filed: January 15, 2004	Art Unit: 2174
For: A SYSTEM AND METHOD FOR SCHEDULING THE EXECUTION OF MODEL COMPONENTS USING MODEL EVENTS	Confirmation No.: 7444

Cesari and McKenna, LLP
88 Black Falcon Avenue
Boston, MA 02210
December 20, 2010

CERTIFICATE OF EFS WEB TRANSMISSION

I hereby certify that the following paper is being EFS WEB transmitted to the Patent and Trademark Office on December 20, 2010.

/Merisa Jakupovic/
Merisa Jakupovic

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPEAL BRIEF

In response to the Notice of Appeal transmitted on June 29, 2010 and the Notice of Panel Decision from Pre-Appeal Brief Review mailed August 19, 2010, Applicant hereby submits this Appeal Brief.

REAL PARTY IN INTEREST

The real party in interest is The MathWorks, Inc. of Natick, Massachusetts.

RELATED APPEALS AND INTERFERENCES

Applicant and its legal representative know of no related appeals or interferences that will directly affect, be directly affected by, or have a bearing on the Board's decision in the present appeal.

STATUS OF CLAIMS

Claims 1-29 and 33-61 are pending. Claims 1-29 and 33-61 stand finally rejected under 35 U.S.C. §103, as set forth in the final Office Action mailed March 30, 2010. The rejection of claims 1-29 and 33-61 is appealed.

A copy of claims 1-29 and 33-61 is attached hereto as an Appendix.

STATUS OF AMENDMENTS

No amendments have been filed since the mailing of the final Office Action on March 30, 2010.

SUMMARY OF CLAIMED SUBJECT MATTER

This summary is set forth in four exemplary embodiments that correspond to independent claims 1, 19, 33, and 51. Discussions about elements and recitations of these claims can be found at least at the cited locations in the Specification and drawings.

Independent claim 1 is directed to a method for controlling the execution of a graphical model. The method includes displaying a view of the model (20), the model (20) having a plurality of time-based components (28, 30), including a user-configurable post component (22) that posts an event (A) when a condition associated with its input

signal (24) is satisfied. One of the time-based components (28) of the model (20) is logically associated with the event (A). The method identifies when the condition is satisfied during execution of the model (20), and posts the event (A) by the post component (22), by informing an event handler. The logically associated component (28) is notified of the occurrence of the event (A), and the occurrence of the event (A) triggers the execution of the time-based component (28). The method further includes executing, during a simulation of the model (20), the logically associated, time-based component (28) in response to being notified of the occurrence of the event (A), as opposed to executing the logically associated component (28) in response to a specific point in time. Specification, p. 7, lines 4-19, and Fig. 2.

Independent claim 19 is directed to a method for controlling the execution of a graphical model. The method includes displaying a view of the model (20, 110), the model (20) having a plurality of time-based components (28, 30), including a user-configurable post component (22) that posts a specified event (A) when a condition associated with its input signal (24) is satisfied. The method identifies when the condition is satisfied during execution of the model (20, 110), and posts the specified event (A) by informing an event handler of an occurrence of the specified event (A). In response to the posting of the specified event (A), the execution of another event (B), referred to as an executing event, is interrupted, and an operation is performed in the model. Specification, p. 7, lines 4-19, p. 13, lines 9-11, and Figs. 2, 6B and 6C.

Independent claim 33 is directed to a physical medium containing computer-executed instructions for controlling the execution of a graphical model. The instructions include instructions for displaying a view of the model (20), the model (20) having a

plurality of time-based components (28, 30), including a user-configurable post component (22) that posts an event (A) when a condition associated with its input signal (24) is satisfied. One of the time-based components (28) of the model (20) is logically associated with the event (A). The instructions also include instructions for identifying when the condition is satisfied during execution of the model (20), and for posting the event (A) by the post component (22), by informing an event handler. The logically associated component (28) is notified of the occurrence of the event (A), and the occurrence of the event (A) triggers the execution of the time-based component (28). The instructions further include instructions for executing, during a simulation of the model (20), the logically associated, time-based component (28) in response to being notified of the occurrence of the event (A), as opposed to executing the logically associated component (28) in response to a specific point in time. Specification, p. 7, lines 4-19, and Fig. 2.

Independent claim 51 is directed to a physical medium containing computer-executed instructions for controlling the execution of a graphical model. The instructions include instructions for displaying a view of the model (20, 110), the model (20) having a plurality of time-based components (28, 30), including a user-configurable post component (22) that posts a specified event (A) when a condition associated with its input signal (24) is satisfied. The instructions include instructions for identifying when the condition is satisfied during execution of the model (20, 110), and posting the specified event (A) by informing an event handler of an occurrence of the specified event (A). In response to the posting of the specified event (A), the execution of another event (B), referred to as an executing event, is interrupted, and an operation is performed in the

model. Specification, p. 7, lines 4-19, p. 13, lines 9-11, and Figs. 2, 6B and 6C.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether independent claims 1, 19, 33, and 51, which satisfy all other conditions of patentability, are unpatentable under 35 U.S.C. §103 on the grounds that they are obvious based on U.S. Patent No. 7,134,109 to Hayles ("Hayles") in view of U.S. Patent No. 5,522,073 to Courant et al. ("Courant") and U.S. Patent No. 6,880,130 to Makowski et al. ("Makowski").

Whether claims 4 and 36, which satisfy all other conditions of patentability, are unpatentable under 35 U.S.C. §103 on the grounds that they are obvious based on Hayles in view of Courant and Makowski.

Whether claims 14 and 46, which satisfy all other conditions of patentability, are unpatentable under 35 U.S.C. §103 on the grounds that they are obvious based on Hayles in view of Courant and Makowski.

Whether claims 20 and 52, which satisfy all other conditions of patentability, are unpatentable under 35 U.S.C. §103 on the grounds that they are obvious based on Hayles in view of Courant and Makowski.

Whether claims 28 and 60, which satisfy all other conditions of patentability, are unpatentable under 35 U.S.C. §103 on the grounds that they are obvious based on Hayles in view of Courant and Makowski.

ARGUMENT

Legal Standard

Section 103(a) states:

> A patent may not be obtained … if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

In rejecting claims under 35 U.S.C. §103, the examiner bears the initial burden of presenting a prima facie case of obviousness. *In re Rijckaert*, 9 F.3d 1531, 1532, 28 U.S.P.Q.2d 1955 (Fed. Cir. 1993). As the Supreme Court recently held, the question of obviousness is resolved on the basis of underlying factual determinations including (1) the scope and content of the prior art, (2) the differences between the claimed subject matter and the prior art, and (3) the level of skill in the art. *KSR Internat'l Co. v. Teleflex Inc.*, 127 S.Ct. 1727, 1734 (2007).

Furthermore, as stated by the Federal Circuit:

> (T)he examiner bears the initial burden … of presenting a prima facie case of unpatentability. … After evidence or argument is submitted by the applicant in response, patentability is determined on the totality of the record, by a preponderance of evidence with due consideration to persuasiveness of argument.

In re Oetiker, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443 (Fed. Cir. 1992).

When determining whether a claim is obvious, an examiner must make a "searching comparison of the claimed invention – including all of its limitations – with the teaching of the prior art." *In re Ochiai*, 71 F.3d 1565, 1572, 37 U.S.P.Q.2d 1127 (Fed. Cir. 1995). Thus, "obviousness requires a suggestion of all limitations in a claim." *CFMT, Inc. v. Yieldup Intern. Corp.*, 349 F.3d 1333, 1342, 68 U.S.P.Q.2d 1940 (Fed. Cir. 2003), *citing In re Royka*, 490 F.2d 981, 985 (CCPA 1974).

The claims do not stand or fall together. Instead Applicant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-headings as required by 37 C.F.R. §41.37(c)(1)(vii).

Scope and Content of the Prior Art

Hayles

Hayles is directed to a system for making it easier for a user to set the timing and triggering attributes of a physical hardware device connected to a computer. An exemplary hardware device is a plug-in data acquisition (DAQ) board. Hayles describes both a prior art method of setting timing and triggering attributes and a purportedly inventive method. In particular, Fig. 9A of Hayles illustrates a prior art graphical program for specifying the timing and triggering attributes of a physical hardware device, namely an E series data acquisition (DAQ) plug-in board from National Instruments Corp. (the assignee of Hayles). Hayles, Col. 21, lines 42-46, and 53-55. The graphical program of Fig. 9A is a LabVIEW graphical program, which is a data flow program. Hayles, Col. 21, lines 55-59, and Col. 9, lines 34-36. The LabVIEW graphical program includes a Single Scan Virtual Instrument (VI) node associated with the E series DAQ card that triggers the E series DAQ board to acquire data. Hayles, Col. 21, lines 59-61.

Fig. 9B of Hayles, which is meant for comparison with Fig. 9A, illustrates how a user can configure the E series DAQ board according to Hayles' purportedly inventive method. In Fig. 9B, a number of diagram sections, such as a Master Timebase section, an Analog Input (AI) Sample Timebase section, etc. are provided. Hayles, Col. 22, lines 12-16. Each of these diagram sections includes user interface (UI) elements that may be

configured by the user. Hayles, Col. 22, lines 16-17. Specifically, the diagram sections include switches that may be used to select a desired clock source, and data entry fields that may be used to specify values. In addition, a selected clock source may be routed to, and used by, other sections, e.g., through a divisor icon that reduces the input clock rate.

In addition to the various diagram sections, Fig. 9B also includes a state diagram (shown in the lower left portion of the figure). Hayles, Col. 22, lines 57-59. The state diagram includes a plurality of nodes that indicate the operation of the E-series DAQ board. Hayles, Col. 22, lines 59-60.

Courant

Courant describes a system for linking together the execution of different software programs stored on a computer. A user selects a first software program, also referred to as a tool, such as a text edit tool. The user also selects a particular "when" event of the first tool. The user then selects a second tool, and a particular "then" operation to be carried out by the second tool upon occurrence of the "when" event by the first tool. Courant, Col. 2, lines 36-46. In this way, the user can customize the interaction of different software tools stored on the computer. An event server (50) serves as a central event distribution mechanism among the software tools of the computer. Courant, Col. 6, lines 36-38. A message connector (56) listens for the occurrence of pre-defined "when" events, and notifies the event server (50) of the occurrence of any "when" events. The event server then issues a request to the associated tool to run its pre-defined "then" operation. Courant, Col. 7, lines 6-12.

Makowski

Makowski, like the prior art described in Hayles, uses a graphical program to

specify the timing and triggering attributes of a physical hardware device. A graphical program may include a first timing and triggering node that provides parameters for configuring a physical hardware device, such as a signal generator or a measurement device. Makowski, Col. 5, lines 31-32, and Col. 3, lines 58-65. A user may specify a particular timing (or triggering) type for the first node, and in response, the first node is replaced with a second node that is especially suited to configuring the hardware device according to the selected timing (or triggering) type. Makowski, Col. 5, lines 47-58.

Level of Skill

Applicant submits that the level of skill in the art is a person having at least a B.S. degree in computer science or electrical engineering, and several years work experience in graphical programming system design.

Differences Between the Claimed Invention and the Cited References

Numerous differences exist between the claimed invention and the cited references, and, as a result of these differences, the obviousness rejection of the claims should be reversed.

Independent Claims 1 and 33

Claim 1 recites in part:

> displaying a view of an executable graphical model with a plurality of executable time-based components, said … model including at least one user-configurable, executable graphical post component having at least one input port for receiving at least one input signal, said … post component being configured to post an event when a condition associated with said at least one input signal … is satisfied;
>
> logically associating at least one executable time-based component with said event; [and]
>
> executing … said at least one executable time-based component in

response to said notifying as opposed to in response to a specific point in time.

No Executable Graphical Model of Time-Based Components.

The cited references either singly or in combination fail to disclose or suggest the claimed "graphical model with a plurality of executable *time-based* components" including a "post component" having an input port for receiving an input signal. Regarding the term "time-based component", the Specification states:

> Simulink™ from The MathWorks, Inc. of Natick, Massachusetts, is an example of a graphical modeling environment, specifically a block diagram environment. Simulink™ allows users to create a pictorial model of a dynamic system. The model consists of a set of symbols, called blocks. Each block can have zero or more input ports, output ports, and states. Each block represents a dynamic system whose inputs, states, and outputs can change continuously and/or discretely *at specific points in time*.

Specification, p. 1, lines 21-27. *See also* Fig. 2, which illustrates a model 20 with a plurality of executable time-based components, e.g., blocks 28, 30, and 32.

The final Office Action cites to Hayles, Col. 17, line 25 to Col. 22, line 20 and Figs. 8 and 9 as purportedly teaching this limitation. *See* final Office Action, at p. 2. This portion of Hayles, which exceeds four columns of text (over 2900 words), describes both the prior art method of setting timing and triggering attributes of a physical hardware device, and Hayles' purportedly inventive method. Neither involves a graphical model of executable *time-based* components. While the prior art method mentioned by Hayles includes an executable graphical program, it lacks any *time-based* components. The diagram sections of Hayles' purportedly inventive method are not graphical programs and, in any event, lack any executable *time-based* components.

More specifically, Fig. 9A of Hayles shows an exemplary graphical program used in the prior art method of configuring the timing and triggering attributes of a physical

hardware device. This diagram is described by Hayles as a LabVIEW graphical program. Hayles, Col. 21, lines 55-56. Hayles also states that LabVIEW is a dataflow programming environment in which the nodes of the diagram execute, not as a function of time, but when the nodes have data on their input ports. Hayles, Col. 9, lines 35-36. Hayles mentions that, in addition to dataflow, graphical programs may execute based on control flow, execution flow, or signal flow, which Hayles identifies as a subset of dataflow. Hayles, Col. 6, lines 36-38 and Col. 12, lines 34-37. Thus, at best, Hayles suggests the use of dataflow, control flow, execution flow, or signal flow graphical programs.

Fig. 9B of Hayles illustrates his purportedly inventive method, which involves the use of a diagram. Hayles, Col. 17, lines 27-41. Unlike the LabVIEW graphical program of Fig. 9A, which is a dataflow graphical program, the diagram of Fig. 9B is not an executable graphical program. Instead, the diagram of Fig. 9B provides the user with a graphical “indication”, i.e., a picture, of the timing and triggering attributes established for the physical hardware device. Hayles, Col. 17, lines 38-39 (“Thus, the diagram may graphically indicate various of the components of the device”). The diagram is divided into a series of diagram sections each having a number of User Interface (UI) elements, such as user-settable switches and data entry fields, that the user may operate to specify the timing and triggering attributes of the physical hardware device. Hayles, Col. 18, lines 1-15, and Fig. 9B.

Once the user has entered the desired values for the various timing and trigger attributes, the attributes are stored in memory, and may be used to configure the physical hardware device. Hayles, Col. 20, lines 5-9. In other words, the diagram of Fig. 9B

receives user-specified values, stores those values, and then applies them to the hardware device. While the user specified values may be applied to the device through program code based on the diagram, the diagram itself is not executable. Hayles, Col. 20, lines 16-18. This fact is demonstrated at least at Col. 20, lines 31-37 of Hayles, which states that a user may select an 'apply' or 'configure' button associated with Hayles' diagram, which invokes a configuration process, resulting in the physical hardware device being configured. If the diagram itself were executable, the user would be provided with a 'run' or 'execute' button. Instead, the diagram of Fig. 9B is simply a pictorial representation of the timing and triggering attributes chosen by the user.

In addition to the settable switches and the data entry fields, Fig. 9B also includes a state diagram. Hayles, Col. 20, lines 38-40. Hayles' state diagram provides another visual indication to the user of the manner by which the timing and triggering attributes of the device have been set. While Hayles mentions that the state diagram may also be implemented as a software program, *see* Col. 20, line 43-46, Hayles nowhere states that the state diagram is itself executable. A fair and proper reading of Hayles limits its state diagram to a non-executable, illustration that is generated as a visual aid to the user. Hayles, at Col. 20, lines 38-40 ("a state diagram may also be displayed, where the state diagram *indicates* effects of modifying the diagram upon operation of the device.") In any event, a state diagram, by definition, does not include any *time-based* components.

In sum, neither Fig. 9A nor Fig. 9B of Hayles discloses or suggests a graphical model having executable *time-based* components, as recited in claim 1. The remaining portions of Hayles are equally deficient. For example, Fig. 10A illustrates another LabVIEW, i.e., dataflow, graphical program. Hayles, Col. 23, lines 18-20. Similarly,

Fig. 10B illustrates another diagram according to the purportedly inventive method. Hayles, Col. 23, lines 44-46.

The other references, Courant and Makowski, fail to disclose or suggest a graphical model of executable *time-based* components. Courant is not a graphical programming reference at all. Instead, with Courant, a user operates drop down menus of a User Interface to select software tools, “when” events, and corresponding “then” operations. *See* Courant, Figs. 7, 8A and 8B. While Courant does state that a user may cause a “then” operation of a software tool to execute in response to elapsed time, Courant, Col. 3, lines 3-5, this cannot be equated with the claimed graphical model of *time-based* components. Courant’s software tools are not graphical symbols, such as blocks, that may be used to build a graphical model. Instead, Courant’s software tools are themselves complete and independent programs that perform pre-defined functions, and are stored in computer memory. Courant, Col. 5, lines 49-51. Exemplary software tools include an edit tool, a build tool, a spell check tool, a debug tool, and a version management tool. Courant, Col. 1, lines 34-35. In addition, Courant appears to follow an event-driven execution model. Courant, Col. 7, lines 17-18, and Col. 9, lines 46-52.

Makowski, like the prior art of Hayles, discloses a dataflow graphical program, but also mentions control flow and execution flow, as other alternatives. Makowski, Col. 14, lines 61-63. Makowski is silent regarding a graphical model of time-based components.

<u>No Post Component</u>.

The final Office Action cites to Hayles as teaching an executable graphical model having a post component, and to Courant as teaching a post component that posts an

event when a condition associated with its input signal is satisfied. *See* final Office Action, at pp. 2-3. Neither reference discloses or suggests the claimed post component.

First, the final Office Action does not point to any particular element of Hayles as disclosing or suggesting the claimed post component. Instead, the final Office Action cites generally to the same four-plus columns of Hayles identified above, i.e., Col. 17 line 25 to Col. 22, line 20. *See* final Office Action, p. 2. As it fails to explain where or how the claimed post component is suggested by Hayles, the final Office Action fails to establish a prima facie case of obviousness.

Second, an examination of Hayles demonstrates that neither the prior art LabVIEW diagram (of Fig. 9A) nor the purportedly inventive diagram (of Fig. 9B) discloses or suggests a post component, as claimed. More specifically, Fig. 9A includes a Single Scan VI node located within an execution loop and, each time the Single Scan VI node executes, data is acquired by the hardware device. Hayles, Col. 21, lines 59-61. There is no mention by Hayles that its Single Scan VI node (or any other node) posts an event when a condition associated with its input signal is satisfied. Regarding the diagram of Fig. 9B, as mentioned above, this diagram is not an executable diagram at all. Furthermore, there is no disclosure or suggestion by Hayles of any post component in Fig. 9B that posts an event in response to a condition of an input signal to the post component being satisfied.

The portion of Courant cited in the final Office Action states:

> According to the present invention, these and other objects and advantages are achieved in a method and apparatus for automating and controlling execution of and modifying the behavior of tools and tool sets in a computer software system that includes a user interface, an operating system and one or more integrated software tools that perform predefined tasks and communicate with

> each other through events passed over an event server. The system is a when/then system, whereby “when” a specified event is detected, “then” a corresponding operation is initiated.
>
> The method includes the steps of determining which tools are available for functionality connection, and presenting the tools to the user for user selection. Once the user selects a tool, the specific functions for the selected tool are presented to the user, so that the user may further select a specific function or event of the selected tool as the “when” event. After the user has selected the specific event that will be the “when” event, the user is presented with a list of tools to select from to represent the action tool for the “then” operation. Once the user selects a tool for the “then” operation, the specific functions for the selected tool are presented to the user, so that the user may further select a specific function of the selected “then” operation tool. The user may also select more than one “then” operation to follow a single “when” event.
>
> After the user has created his routine, he may then name it, save it, and enable or disable it.

Courant, Col. 2, lines 20-50.

As shown, this excerpt fails to disclose or suggest any component of a graphical model, let alone a post component that posts a specified event when a signal received at the post component satisfies a condition. Instead, with Courant, a software tool, issues a “when” event when a specific function or event selected by the user occurs. There is no disclosure or suggestion by Courant of somehow posting an event when a signal received at a component of a graphical model satisfies a condition.

Courant’s software tools cannot fairly be equated with the claimed post component of a graphical model. First, Courant’s software tools are not graphical blocks of an executable graphical model. Instead, they are complete and independent procedures stored on a computer memory. They are not graphical symbols that may be placed on a displayed workspace and arranged as a graphical model. Second, Courant’s software tools have no input ports, and thus cannot receive input signals, as depicted in a graphical

model.

No Logical Association of a Time-Based Component with an Event.

The final Office Action again cites the same four-plus columns of Hayles, i.e., Col. 17, line 25 to Col. 22, line 20, as purportedly teaching the claimed logically associating a time-based component of a graphical model with an event. *See* final Office Action, p. 2. As discussed above, this section of Hayles fails to disclose or suggest a graphical model of time-based components. Instead, Hayles discloses a prior art dataflow graphical program, and a purportedly inventive, non-executable diagram of user-settable switches and data entry fields for configuring the timing and triggering attributes of a physical hardware device. Hayles is silent regarding any association of a time-based component of a graphical model with an event.

The other references are equally deficient. With Courant, an operation or event of a first software tool is designated as a "when" event, and an operation of a second software tool is designated as a "then" operation. Thereafter, in response to the "when" event of the first software tool, the "then" operation of the second software tool is performed. Courant's second software tool is not the same as the claimed *time-based* component of a graphical model. More specifically, Courant's software tools are complete and independent procedures stored in a computer memory. Courant, Col. 5, lines 49-51. Courant's software tools are not graphical symbols, such as blocks, that may be used to build a graphical model. In addition, Courant appears to follow an execution flow design. Courant, Col. 7, lines 17-18, and Col. 9, lines 46-52.

<u>No Execution of a Time-Based Component of an Executable Graphical Model in Response to the Notification of an Event, As Opposed to a Specific Point in Time</u>.

As noted above, claim 1 further recites:

> executing … said at least one executable time-based component in response to said notifying as opposed to in response to a specific point in time.

According to this limitation, a time-based component of an executable graphical model executes, not as a function of time, but in response to the occurrence of event (posted by the post component). The final Office Action again cites to the same four-plus columns of Hayles, i.e., Col. 17, line 25 to Col. 22, line 20, as disclosing this limitation. *See* final Office Action, at p. 3. Additionally, the final Office Action cites to Makowski at Col. 13, lines 25-40 as "teaching executing within said graphical modeling environment during a simulation of said executable graphical model." *See* final Office Action, at p. 4. A review of the cited references demonstrates that they fail to disclose or suggest this limitation.

As discussed above, Hayles fails to disclose or suggest an executable graphical model having *time-based* components. Instead, Hayles describes a prior art approach that utilizes a dataflow diagram, in which the nodes execute, not as a function of time, but in response to data being available to the nodes. Hayles, at Col. 21, lines 55-56, and Fig. 9A. Hayles also discloses a pictorial, non-executable diagram that indicates how the timing and triggering attributes of a physical hardware device have been set by a user, e.g., by setting switches, and entering values into data fields. Hayles, at Col. 22, lines 10-20, and Fig. 9B. At no point does Hayles disclose or suggest a *time-based* component of an executable graphical model that is made to execute in response to the occurrence of an

event, as opposed to a point in time.

Makowski is similarly deficient. First, Makowski, like Hayles, discloses a dataflow, control flow, or execution flow graphical program. Makowski, Col. 14, lines 61-63. Makowski does not disclose or suggest an executable graphical model of *time-based* components. Second, Makowski does not disclose or teach executing a time-based component of a graphical model in response to an event as opposed to executing the time-based component as a specific point in time.

The cited excerpt from Makowski describes how a user can create a graphical program on a computer, and use it with a hardware device that is located remotely from the computer running the graphical program. The excerpt does not disclose or suggest executing a *time-based* component of a graphical model in response to an event, instead of at a point in time, as claimed.

Courant too is deficient. First, Courant is not a graphical programming reference. There is no teaching or suggestion by Courant of an executable graphical model. Second, Courant is completely silent regarding the execution of a time-based component of a model in response to an event, as opposed to a specific point in time.

In sum, because the cited references fail to disclose or suggest at least:

- an executable graphical model of *time-based* components,
- a post component of the graphical model that posts an event when a condition associated with its input signal is satisfied,
- logically associating a time-based component of the graphical model with the event, or
- executing the associated *time-based* component in response to the event as opposed to a specific point in time,

the rejection of claim 1 under §103 should be reversed. Independent claim 33 includes similar limitations as claim 1, and thus, it too is non-obvious over the cited references, and the rejection of claim 33 also should be reversed.

Insufficient Reason to Combine

The final Office Action states that one skilled in the art would be motivated to combine Hayles and Courant "to give the user greater flexibility to select specific functions related to the event." See final Office Action, p. 3. However, there does not appear to be any limitation on the selection of "then" operations by Courant. That is, with Courant, after selecting a desired "when" event, the system already presents all of the available "then" events. Courant, Col. 2, lines 39-46. Furthermore, the final Office Action fails to explain how a combination of Hayles and Courant, even assuming such a combination were possible which Applicant submits it is not, would give "greater flexibility".

The final Office Action also states that one skilled in the art would be motivated to combine Makowski with Hayles "to allow user to simulate the industrial environment." See final Office Action, p. 4. However, Hayles explicitly states that its approach may be used in "industrial automation". Hayles, Col. 13, lines 56-65. Given this existing capability of Hayles, one skilled in the art would not be motivated to combine Hayles with another reference, at least not for the reason stated in the final Office Action. That is, one skilled in the art would not be motivated to combine Hayles with another reference to achieve a capability that Hayles already performs.

Accordingly, Applicant submits that the final Office Action fails to present a sufficient reason to combine the cited references.

Independent Claims 19 and 51

Independent method claim 19 recites, in part:

> displaying … an executable model with a plurality of executable time-based components, said model including [a] post component having at least one input port for receiving at least one input signal, said … post component being configured to post a specified event when a condition associated with said … input signal … is satisfied
>
> interrupting execution of an executing event in response to said posting of said specified event; and
>
> performing an operation in said executable model within said graphical modeling environment in response to said posting of said specified event.

The final Office Action does not provide any independent analysis of claim 19. Instead, the final Office Action rejects claim 19 "under the same rationale as claim 1." *See* final Office Action, p. 7. Claim 19, however, includes additional limitations not found in claim 1, such as the "interrupting" and "performing" limitations. Accordingly, Applicant submits that the rejection of claim 19 is deficient as a matter of law. *See, e.g., In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031 (Fed. Cir. 1994) (stating that the "Patent and Trademark Office (PTO) must consider all claim limitations when determining patentability of an invention over the prior art.") Furthermore, numerous differences exist between claim 19 and the cited references.

<u>No Executable Graphical Model of Time-Based Components or Post Component</u>.

The cited references (Hayles, Courant and Makowski) fail to disclose or suggest either an executable model of *time-based* components, or a post component that posts an event when a condition associated with an input signal to the post component is satisfied. This limitation is similar to the limitation of claim 1, and the following argument repeats

the above argument presented in connection with claims 1 and 33.

The cited references either singly or in combination fail to disclose or suggest the claimed "graphical model with a plurality of executable *time-based* components" including a "post component" having an input port for receiving an input signal. Regarding the term "time-based component", the Specification states:

> Simulink™ from The MathWorks, Inc. of Natick, Massachusetts, is an example of a graphical modeling environment, specifically a block diagram environment. Simulink™ allows users to create a pictorial model of a dynamic system. The model consists of a set of symbols, called blocks. Each block can have zero or more input ports, output ports, and states. Each block represents a dynamic system whose inputs, states, and outputs can change continuously and/or discretely *at specific points in time*.

Specification, p. 1, lines 21-27. *See also* Fig. 2, which illustrates a model 20 with a plurality of executable time-based components, e.g., blocks 28, 30, and 32.

The final Office Action cites to Hayles, Col. 17, line 25 to Col. 22, line 20 and Figs. 8 and 9 as purportedly teaching this limitation, as appearing in claim 1. *See* final Office Action, at p. 2. This portion of Hayles describes both the prior art method of setting timing and triggering attributes of a physical hardware device, and Hayles' purportedly inventive method. Neither involves a graphical model of executable *time-based* components. While the prior art method mentioned by Hayles includes an executable graphical program, it lacks any *time-based* components. The diagram sections of Hayles' purportedly inventive method are not graphical programs and, in any event, lack any executable *time-based* components.

More specifically, Fig. 9A of Hayles shows an exemplary graphical program used in the prior art method of configuring the timing and triggering attributes of a physical hardware device. This diagram is described by Hayles as a LabVIEW graphical program.

Hayles, Col. 21, lines 55-56. Hayles also states that LabVIEW is a dataflow programming environment in which the nodes of the diagram execute, not as a function of time, but when the nodes have data on their input ports. Hayles, Col. 9, lines 35-36. Hayles mentions that, in addition to dataflow, graphical programs may execute based on control flow, execution flow, or signal flow, which Hayles identifies as a subset of dataflow. Hayles, Col. 6, lines 36-38 and Col. 12, lines 34-37. Thus, at best, Hayles suggests the use of dataflow, control flow, execution flow, or signal flow graphical programs.

Fig. 9B of Hayles illustrates his purportedly inventive method, which involves the use of a diagram. Hayles, Col. 17, lines 27-41. Unlike the LabVIEW graphical program of Fig. 9A, which is a dataflow graphical program, the diagram of Fig. 9B is not an executable graphical program. Instead, the diagram of Fig. 9B provides the user with a graphical "indication", i.e., a picture, of the timing and triggering attributes established for the physical hardware device. Hayles, Col. 17, lines 38-39 ("Thus, the diagram may graphically indicate various of the components of the device"). The diagram is divided into a series of diagram sections each having a number of User Interface (UI) elements, such as user-settable switches and data entry fields, that the user may operate to specify the timing and triggering attributes of the device. Hayles, Col. 18, lines 1-15, and Fig. 9B.

Once the user has entered the desired values for the various timing and trigger attributes, the attributes are stored in memory, and may be used to configure the physical hardware device. Hayles, Col. 20, lines 5-9. In other words, the diagram of Fig. 9B receives user-specified values, stores those values, and then applies them to the hardware

device. While the user specified values may be applied to the device through program code based on the diagram, the diagram itself is not executable. Hayles, Col. 20, lines 16-18. This fact is demonstrated at least at Col. 20, lines 31-37 of Hayles, which states that a user may select an 'apply' or 'configure' button associated with Hayles' diagram, which invokes a configuration process, resulting in the physical hardware device being configured. If the diagram itself were executable, the user would be provided with a 'run' or 'execute' button. Instead, the diagram of Fig. 9B is simply a pictorial representation of the timing and triggering attributes chosen by the user.

In addition to the settable switches and the data entry fields, Fig. 9B also includes a state diagram. Hayles, Col. 20, lines 38-40. Hayles' state diagram provides another visual indication to the user of the manner by which the timing and triggering attributes of the device have been set. While Hayles mentions that the state diagram may also be implemented as a software program, *see* Col. 20, line 43-46, Hayles nowhere states that the state diagram is itself executable. A fair and proper reading of Hayles limits its state diagram to a non-executable, illustration that is generated as a visual aid to the user. Hayles, at Col. 20, lines 38-40 ("a state diagram may also be displayed, where the state diagram *indicates* effects of modifying the diagram upon operation of the device.") In any event, a state diagram, by definition, does not include any *time-based* components.

In sum, neither Fig. 9A nor Fig. 9B of Hayles discloses or suggests an executable model having executable *time-based* components, as recited in claim 19. The remaining portions of Hayles are equally deficient. For example, Fig. 10A illustrates another LabVIEW, i.e., dataflow, graphical program. Hayles, Col. 23, lines 18-20. Similarly, Fig. 10B illustrates another diagram according to the purportedly inventive method.

Hayles, Col. 23, lines 44-46.

The other references, Courant and Makowski, fail to disclose or suggest a graphical model of executable *time-based* components. Courant is not a graphical programming reference at all. Instead, with Courant, a user operates drop down menus of a User Interface to select software tools, "when" events, and corresponding "then" operations. *See* Courant, Figs. 7, 8A and 8B. While Courant does state that a user may cause a "then" operation of a software tool to execute in response to elapsed time, Courant, Col. 3, lines 3-5, this cannot be equated with the claimed graphical model of *time-based* components. Courant's software tools are not graphical symbols, such as blocks, that may be used to build a graphical model. Instead, Courant's software tools are themselves complete and independent programs that perform pre-defined functions, and are stored in computer memory. Courant, Col. 5, lines 49-51. Exemplary software tools include an edit tool, a build tool, a spell check tool, a debug tool, and a version management tool. Courant, Col. 1, lines 34-35. In addition, Courant appears to follow an event-driven execution model. Courant, Col. 7, lines 17-18, and Col. 9, lines 46-52.

Makowski, like the prior art of Hayles, discloses a dataflow graphical program, but also mentions control flow and execution flow, as other alternatives. Makowski, Col. 14, lines 61-63. Makowski is silent regarding a graphical model of time-based components.

With regard to claim 1, the final Office Action cites to Hayles as teaching an executable graphical model having a post component, and to Courant as teaching a post component that posts an event when a condition associated with its input signal is satisfied. *See* final Office Action, at pp. 2-3. Neither reference discloses or suggests the

claimed post component recited in claim 19.

First, the final Office Action does not point to any particular element of Hayles as disclosing or suggesting the claimed post component. Instead, the final Office Action cites generally to the same four-plus columns of Hayles identified above, i.e., Col. 17 line 25 to Col. 22, line 20. *See* final Office Action, p. 2. As it fails to explain where or how the claimed post component is suggested by Hayles, the final Office Action fails to establish a prima facie case of obviousness.

Second, an examination of Hayles demonstrates that neither the prior art LabVIEW diagram (of Fig. 9A) nor the purportedly inventive diagram (of Fig. 9B) discloses or suggests a post component, as claimed. More specifically, Fig. 9A includes a Single Scan VI node located within an execution loop and, each time the Single Scan VI node executes, data is acquired by the hardware device. Hayles, Col. 21, lines 59-61. There is no mention by Hayles that its Single Scan VI node (or any other node) posts an event when a condition associated with its input signal is satisfied. Regarding the diagram of Fig. 9B, as mentioned above, this diagram is not an executable diagram at all. Furthermore, there is no disclosure or suggestion by Hayles of any post component in Fig. 9B that posts an event in response to a condition of an input signal to the post component being satisfied.

The portion of Courant cited in the final Office Action in connection with claim 1 states:

> According to the present invention, these and other objects and advantages are achieved in a method and apparatus for automating and controlling execution of and modifying the behavior of tools and tool sets in a computer software system that includes a user interface, an operating system and one or more integrated software tools that perform predefined tasks and communicate with

> each other through events passed over an event server. The system is a when/then system, whereby "when" a specified event is detected, "then" a corresponding operation is initiated.
>
> The method includes the steps of determining which tools are available for functionality connection, and presenting the tools to the user for user selection. Once the user selects a tool, the specific functions for the selected tool are presented to the user, so that the user may further select a specific function or event of the selected tool as the "when" event. After the user has selected the specific event that will be the "when" event, the user is presented with a list of tools to select from to represent the action tool for the "then" operation. Once the user selects a tool for the "then" operation, the specific functions for the selected tool are presented to the user, so that the user may further select a specific function of the selected "then" operation tool. The user may also select more than one "then" operation to follow a single "when" event.
>
> After the user has created his routine, he may then name it, save it, and enable or disable it.

Courant, Col. 2, lines 20-50.

As shown, this excerpt fails to disclose or suggest any component of a graphical model, let alone a post component that posts a specified event when a signal received at the post component satisfies a condition. Instead, with Courant, a software tool, issues a "when" event when a specific function or event selected by the user occurs. There is no disclosure or suggestion by Courant of somehow posting an event when a signal received at a component of a graphical model satisfies a condition.

Courant's software tools cannot fairly be equated with the claimed post component. First, Courant's software tools are not graphical blocks of an executable model. Instead, they are complete and independent procedures stored on a computer memory. They are not graphical symbols that may be placed on a displayed workspace and arranged as a graphical model. Second, Courant's software tools have no input ports, and thus cannot receive input signals, as depicted in a graphical model.

No Interruption of an Executing Event, or Performance of an Operation in a Graphical Model in Response to the Occurrence of the Specified Event.

As shown above, claim 19 also recites interrupting the execution of an executing event, and performing an operation within a graphical model in response to the posting of the specified event by the post component. None of the cited references discloses or suggests these claim limitations.

Hayles does not disclose or suggest the interruption of an executing event, or the performance of an operation within an executable model in response to the occurrence of a specified event. Hayles discloses two types of diagrams: a prior art, dataflow graphical program (see Fig. 9A), and a visual indication of user-settable switches and data entry fields (Fig. 9B) for configuring the timing and triggering attributes of a physical hardware device. The diagram of Fig. 9B also includes a state diagram that provides another visual indication to the user of the timing and triggering attributes chosen for the physical device. However, there is no disclosure or suggestion by Hayles of interrupting an executing event and performing an operation in a model of time-based components in response to the posting of a specified event by a post component of the model.

Courant and Makowski fail to overcome the deficiencies in Hayles.

Courant does not disclose or suggest interrupting an executing event, or performing an operation in a graphical model in response to the posting of a specified event by a post component of the model. First, Courant is completely silent regarding graphical models. Consequently, Courant fails to disclose or suggest the performance of an operation within a graphical model. Furthermore, there is no mention by Courant of interrupting the execution of an executing event. For example, there is no mention by

Courant of somehow interrupting its "when" event. Indeed, the term "interrupt" nowhere appears in Courant.

Similarly, Makowski does not disclose or suggest interrupting an executing event or performing an operation within a graphical model in response to the posting of an event by a post component.

Because the final Office Action fails to explain where or how the limitations of claim 19 are shown in the cited references, the final Office Action fails to establish a prima facie case obviousness. In addition, a review of the cited references shows that they fail to disclose or suggest at least:

- displaying an executable model with a plurality of time-based components, including a post component that posts an event when a condition associated with its input signal is satisfied,
- interrupting execution of an executing event in response to the posting of the specified event, or
- performing an operation in the model in response to the posting of the specified event.

Accordingly, the rejection of independent claim 19 should be reversed.

Independent claim 51 includes similar limitations as claim 19 and thus, the rejection of claim 51 also should be reversed.

<u>Insufficient Reason to Combine</u>

The final Office Action fails to articulate a separate reason to combine the cited references in support of the rejection of claim 19. Instead, the final Office Action simply states that "claim 19 … is rejected under the same rationale as claim 1." *See* final Office

Action, p. 7.

With regard to claim 1, the final Office Action states that one skilled in the art would be motivated to combine Hayles and Courant "to give the user greater flexibility to select specific functions related to the event." See final Office Action, p. 3. However, there does not appear to be any limitation on the selection of "then" operations by Courant. That is, with Courant, after selecting a desired "when" event, the system already presents all of the available "then" events. Courant, Col. 2, lines 39-46. Furthermore, the final Office Action fails to explain how a combination of Hayles and Courant, even assuming such a combination were possible which Applicant submits it is not, would give "greater flexibility".

The final Office Action also states, again with regard to claim 1, that one skilled in the art would be motivated to combine Makowski with Hayles "to allow user to simulate the industrial environment." See final Office Action, p. 4. However, Hayles explicitly states that its approach may be used in "industrial automation". Hayles, Col. 13, lines 56-65. Given this existing capability of Hayles, one skilled in the art would not be motivated to combine Hayles with another reference, at least not for the reason stated in the final Office Action. That is, one skilled in the art would not be motivated to combine Hayles with another reference to achieve a capability that Hayles already performs.

Accordingly, Applicant submits that the final Office Action fails to present a sufficient reason to combine the cited references.

Dependent Claims 4 and 36

Claim 4 depends from claim 1 and recites:

setting a sample time for an initial execution of at least one executable time-based component to be said occurrence of said specified event.

Regarding the term "sample time", the Specification states as follows:

> Simulink™ sample rates provide a mechanism for specifying how often components of a model execute. [p. 2, lines 16-17]
>
> An event's 'scope' is the scope of the workspace within which it exists. If a workspace is associated with a subsystem or model, the scope of the event is that subsystem or model. Blocks may only specify their sample time as an event when that event is in scope from that block. [p. 8, lines 4-7]

In rejecting claim 4, the final Office Action again cites to the same four-plus columns of Hayles, i.e., Col. 17, line 25 to Col. 22, line 20 and Figs. 8 and 9. *See* final Office Action, at p. 4.

Hayles fails to disclose or suggest the setting of a sample time of a *time-based* component of a graphical model to any value or parameter. Consequently, Hayles also fails to disclose or suggest the setting of a sample time of a *time-based* component to the occurrence of an event. First, Hayles does not disclose a graphical model of time-based components. Instead, Hayles describes a dataflow graphical program. Because the nodes of a dataflow graphical program execute in response to the availability of data, there is no concept of a "sample time" for any of the nodes of Hayles' dataflow program. Not surprisingly, the term "sample time" nowhere appears in Hayles.

Hayles does mention a "sample clock" that is used by the physical hardware device to determine when to acquire data samples. Hayles, Col. 22, lines 46-49. Hayles' "sample clock", however, cannot be equated with the claimed "sample time" for several reasons. First, Hayles' "sample clock" is a component of the physical hardware device being configured. Hayles, Col. 22, lines 10-17. The "sample clock" is not associated

with a *time-based* component of a graphical model. Second, Hayles' "sample clock" is not set to the occurrence of some event. Instead, the user operates one of the switches and one or more of the data entry fields in order to specify a desired rate for the sample clock. Hayles, Fig. 9B.

The other cited references fail to overcome the deficiencies of Hayles.

Neither Courant nor Makowski discloses or suggests a "sample time" of a *time-based* component of a graphical model. Courant does not disclose graphical models at all. Makowski discloses dataflow models in which nodes of the graphical program execute when data is available. Makowski is completely silent regarding setting a sample time for the initial execution of a time-based component to be the occurrence of an event.

Because the final Office Action fails to explain where or how the limitations of claim 4 are shown in the cited references, the final Office Action fails to establish a prima facie case obviousness. In addition, a review of the cited references shows that they fail to disclose or suggest the limitations of claim 4. Accordingly, the rejection of claim 4 should be reversed.

Claim 36, which depends from claim 33, includes similar limitations as claim 4. Accordingly, the rejection of claim 36 also should be reversed.

Dependent Claims 14 and 46

Claim 14, which depends from claim 1, recites:

> wherein an execution scope of said specified event for which said execution of said graphical model is being monitored is restricted to a portion of said graphical model.

Once again, the final Office Action cites to the same four-plus columns of Hayles, i.e., Col. 17, line 25 to Col. 22, line 20 and Figs. 8 and 9, as allegedly disclosing this

limitation. *See* final Office Action, at p. 6. Hayles does not disclose or suggest limiting the execution scope of an event to a portion of a graphical model. Indeed, Hayles is completely silent about somehow restricting an execution scope of an event.

The other references are also deficient. While Courant mentions the use of "when" events, and "then" operations, Courant is silent about restricting an execution scope of an event to a portion of a graphical model. Makowski likewise provides no disclosure or suggestion of the notion of restricting the execution scope of an event.

Because the final Office Action fails to explain where or how the limitations of claim 14 are shown in the cited references, the final Office Action fails to establish a prima facie case obviousness. In addition, a review of the cited references shows that they fail to disclose or suggest the limitations of claim 14. Accordingly, the rejection of claim 14 should be reversed.

Claim 46, which depends from claim 33, includes a similar limitation as claim 14. Accordingly, the rejection of claim 46 also should be reversed.

Dependent Claims 20 and 52

Claim 20, which depends from claim 19, recites:

> wherein said specified event is treated as a normal event and further comprising:
>
> resuming execution of said interrupted event.

Here too, the final Office Action cites to Hayles, Col. 17, line 25 to Col. 22, line 20 and Figs. 8 and 9, as allegedly disclosing this limitation. *See* final Office Action, at p. 8. Hayles fails to disclose or suggest treating an event as a "normal" event, or resuming an interrupted event. As noted above, there is no mention by Hayles of an event that is

interrupted. There is likewise no mention by Hayles of resuming execution of any interrupted event.

The other cited references are also deficient. Neither Courant nor Makowski disclose or suggest resuming execution of an interrupted event.

Because the final Office Action fails to explain where or how the limitation of claim 20 is shown in the cited references, the final Office Action fails to establish a prima facie case of obviousness. Furthermore, a review of the cited references demonstrates that they fail to disclose or suggest the limitation of claim 20. Accordingly, the rejection of claim 20 should be reversed.

Claim 52, which depends from claim 51, includes a similar limitation. Accordingly, the rejection of claim 52 also should be reversed.

Dependent Claims 28 and 60

Claim 28 depends from claim 19, and recites:

> wherein said operation is controlled by an order of execution indicated in a branch priority block.

Regarding the term "branch priority block", the Specification states:

> FIG. 7B shows the placement of a "Branch Priority Block" 170 that specifies that blocks in the lower branch should execute prior to blocks in the upper branch. FIG. 7C indicates the execution order dictated by the Branch Priority Block 170 of FIG. 7B. The number "1" (172) in the Branch Priority Block 170 indicates that the lower branches should execute first. The number "2" in the Branch Priority Block 170 indicates that the upper branch should execute second.

Specification, at p. 15, lines 15-20.

Here again, the final Office Action cites to the same excerpt of Hayles, i.e., Col. 17, line 25 to Col. 22, line 20. *See* final Office Action, p. 10.

A review of Hayles demonstrates that it fails to disclose or suggest a branch priority block, or the use of such a block to control an operation in a graphical model in response to an event.

The other cited references fail to overcome the deficiency of Hayles. Courant is not a graphical modeling reference, and thus it provides no disclosure or suggestion of a branch priority block. Makowski discloses the use of a dataflow diagram, but Makowski is silent regarding a branch priority block, or controlling an operation by a model by a branch priority block.

Because the final Office Action fails to explain where or how the limitation of claim 28 is shown in the cited references, the final Office Action fails to establish a prima facie case of obviousness. Furthermore, a review of the cited references demonstrates that they fail to disclose or suggest either a branch priority block, or controlling the operation of a graphical model with such a block. Accordingly, the rejection of claim 28 should be reversed.

Claim 60, which depends from claim 51, includes a similar limitation. Accordingly, the rejection of claim 60 also should be reversed.

CONCLUSION

Applicant respectfully submits that the claims are allowable over the art of record. Accordingly, Applicant requests that the rejection of all claims be reversed.

Respectfully submitted,

/Michael R. Reinemann/

Michael R. Reinemann
Reg. No. 38,280
CESARI AND MCKENNA, LLP
88 BLACK FALCON AVENUE
BOSTON, MA 02210
Telephone: (617) 951-2500
Facsimile: (617) 951-3927

CLAIMS APPENDIX

1. A method for controlling model execution in a graphical modeling environment, said method comprising:

displaying a view of an executable graphical model with a plurality of executable time-based components, said executable graphical model including at least one user-configurable, executable graphical post component having at least one input port for receiving at least one input signal, said executable graphical post component being configured to post an event when a condition associated with said at least one input signal of said executable graphical post component is satisfied;

logically associating at least one executable time-based component with said event;

identifying when said condition is satisfied during execution of said executable graphical model;

posting, using said executable graphical post component, said event by informing an event handler of an occurrence of said event in said graphical modeling environment;

notifying said at least one executable time-based component that is logically associated with said event of said occurrence of said event, said occurrence of said event triggering an execution of said at least one executable time-based component; and

executing, within said graphical modeling environment, during a simulation of said executable graphical model, said at least one executable time-based component in response to said notifying as opposed to in response to a specific point in time.

2. The method of claim 1, further comprising:

registering said at least one executable time-based component with said event handler.

3. The method of claim 1 wherein said executable graphical post component is a block or label.

4. The method of claim 1, further comprising:

setting a sample time for an initial execution of at least one executable time-based component to be said occurrence of said specified event.

5. The method of claim 4, further comprising:

propagating said sample time to at least one other executable time-based component in said graphical model, said at least one other executable time-based component configured to inherit a sample rate.

6. The method of claim 4, further comprising:

setting a sample time of a plurality of non-contiguous executable time-based components in said graphical model to be said occurrence of said event.

7. The method of claim 6 wherein said sample time for said plurality of non-contiguous executable time-based components is set without adjusting visible connections between said executable time-based components displayed in said view.

8. The method of claim 4, further comprising:

indicating with an event ID in said view that said sample time of said at least one executable time-based component is set to said event.

9. The method of claim 4 wherein said event is an explicit event set by a user.

10. The method of claim 4 wherein said event is an implicit event caused by said execution of said graphical model.

11. The method of claim 10 wherein said implicit event is one of power-up, power-down and initialization.

12. The method of claim 10 wherein said implicit event corresponds to one of enabling and disabling of a subsystem.

13. The method of claim 2, further comprising:

indicating which event an executable time-based component receives with a user-configurable color in said view.

14. The method of claim 1, wherein an execution scope of said specified event for which said execution of said graphical model is being monitored is restricted to a portion of said graphical model.

15. The method of claim 1 wherein each event in said graphical model maps on a one-to-one basis to an event handler, said event handler being a function.

16. The method of claim 15 wherein said function is inlined.

17. The method of claim 1 wherein a branch priority executable time-based component indicates an order of execution among at least two branches of said executable time-based components in response to said notifying.

18. The method of claim 1 wherein more than one executable time-based component group executes in response to said notifying, said executable time-based component groups being a user selected grouping of executable time-based components, said order of execution of said executable time-based component groups specified by a user.

19. A method for controlling model execution in a graphical modeling environment, said method comprising:

displaying a view of an executable model with a plurality of executable time-based components, said model including at least one user-configurable, executable graphical post component having at least one input port for receiving at least one input signal, said graphical post component being configured to post a specified event when a condition associated with said at least one input signal of said executable graphical post component is satisfied;

identifying when said condition is satisfied during said execution of said executable model, said execution of said executable model including running a simulation of said executable model within said graphical modeling environment;

posting, using said executable graphical post component, said specified event by informing an event handler of an occurrence of said specified event in said graphical modeling environment;

interrupting execution of an executing event in response to said posting of said specified event; and

performing an operation in said executable model within said graphical modeling environment in response to said posting of said specified event.

20. The method of claim 19 wherein said specified event is treated as a normal event and further comprising:

resuming execution of said interrupted event.

21. The method of claim 19 wherein said specified event is treated as an exception event and further comprising:

returning control of said execution of said model to a calling process which called said interrupted executing event without resuming execution of said interrupted event.

22. The method of claim 19 wherein said specified event is specified using an instantiated event object.

23. The method of claim 22 wherein said event is an explicit event.

24. The method of claim 22 wherein said event is an implicit event.

25. The method of claim 22 wherein said event object is associated with a task object, said task object corresponding to an operating system task.

26. The method of claim 25 wherein said task object has at least one of a specified execution rate and priority.

27. The method of claim 26 wherein at least two events with different tasks are executing in a model and further comprising:

using event transition components to schedule said execution of time-based components associated with said at least two events, said event transition components separating said execution of said time-based components associated with said at least two events.

28. The method of claim 19 wherein said operation is controlled by an order of execution indicated in a branch priority block.

29. The method of claim 19 wherein said operation is said execution of more than one executable time-based component groups, said executable time-based component

groups being a user selected grouping of executable time-based components, said order of execution of said executable time-based component groups specified by a user.

33. A physical computer-readable medium holding computer-executable instructions for controlling model execution in a graphical modeling environment, said instructions comprising:

one or more instructions for displaying a view of an executable graphical model with a plurality of executable time-based components, said executable graphical model including at least one user-configurable, executable graphical post component having at least one input port for receiving at least one input signal, said executable graphical post component being configured to post an event when a condition associated with said at least one input signal of said executable graphical post component is satisfied;

one or more instructions for logically associating at least one executable time-based component with said event;

one or more instructions for identifying when said condition is satisfied during said execution of said executable graphical model;

one or more instructions for posting, using said executable graphical post component, said event by informing an event handler of an occurrence of said event in said modeling environment;

one or more instructions for notifying said at least one executable time-based component that is logically associated with said event of said occurrence of said event, said occurrence of said event triggering an execution of said at least one executable time-based component; and

one or more instructions for executing, within said graphical modeling environment, during a simulation of said executable graphical model, said at least one executable time-based component in response to said notifying as opposed to in response to a specific point in time.

34. The medium of claim 33, wherein said instructions further comprise:

one or more instructions for registering said at least one executable time-based component with said event handler.

35. The medium of claim 33, wherein said executable graphical post component is a block or label.

36. The medium of claim 33, wherein said instructions further comprise:

one or more instructions for setting a sample time for an initial execution of at least one executable time-based component to be said occurrence of said specified event.

37. The medium of claim 36, wherein said instructions further comprise:

one or more instructions for propagating said sample time to at least one other executable time-based component in said graphical model, said at least one other executable time-based component configured to inherit a sample rate.

38. The medium of claim 36, wherein said instructions further comprise:

one or more instructions for setting a sample time of a plurality of non-contiguous executable time-based components in said graphical model to be said occurrence of said event.

39. The medium of claim 38 wherein said sample time for said plurality of non-contiguous executable time-based components is set without adjusting visible connections between executable time-based components displayed in said view.

40. The medium of claim 36, wherein said instructions further comprise:

one or more instructions for indicating with an event ID in said view that said sample time of said at least one executable time-based component is set to said event.

41. The medium of claim 36 wherein said event is an explicit event set by a user.

42. The medium of claim 36 wherein said event is an implicit event caused by said execution of said graphical model.

43. The medium of claim 42 wherein said implicit event is one of power-up, power-down and initialization.

44. The medium of claim 42 wherein said implicit event corresponds to one of said enabling and disabling of a subsystem.

45. The medium of claim 34, wherein said instructions further comprise:

one or more instructions for indicating which event a executable time-based component receives with a user-configurable color in said view.

46. The medium of claim 33, wherein an execution scope of said specified event for which said execution of said graphical model is being monitored is restricted to a portion of said graphical model.

47. The medium of claim 33 wherein each event in said graphical model maps on a one-to-one basis to an event handler, said event handler being a function.

48. The medium of claim 47 wherein said function is inlined.

49. The medium of claim 33 wherein a branch priority block indicates an order of execution among at least two branches of executable time-based components in response to said notifying.

50. The medium of claim 33 wherein more than one executable time-based component groups execute in response to said notifying, said executable time-based component groups being a user selected grouping of the executable time-based components, said order of execution of said executable time-based component groups specified by a user.

51. A physical computer-readable medium holding computer-executable instructions for controlling model execution, said instructions comprising:

one or more instructions for displaying a view of an executable model with a plurality of executable time-based components, said model including at least one user-configurable, executable graphical post component having at least one input port for receiving at least one input signal, said post component being configured to post a specified event when a condition associated with said at least one input signal of said executable graphical post component is satisfied;

one or more instructions for identifying said satisfaction of said specified condition during said execution of said executable model, said execution of said executable model including running a simulation of said executable model within a graphical modeling environment;

one or more instructions for posting, using said executable graphical post component, said specified event by informing an event handler of an occurrence of said specified event in said graphical modeling environment;

one or more instructions for interrupting execution of an executing event in response to said posting of said specified event; and

one or more instructions for performing an operation in said executable model within said graphical modeling environment in response to said posting of said specified event.

52. The medium of claim 51 wherein said specified event is treated as a normal event and wherein said instructions further comprise:

one or more instructions for resuming execution of said interrupted event.

53. The medium of claim 51 wherein said specified event is treated as an exception event and wherein said instructions further comprise:

one or more instructions for returning control of said execution of said model to a calling process which called said interrupted executing event without resuming execution of said interrupted event.

54. The medium of claim 51 wherein said specified event is specified using an instantiated event object.

55. The medium of claim 54 wherein said event is an explicit event.

56. The medium of claim 54 wherein said event is an implicit event.

57. The medium of claim 54 wherein said event object is associated with a task object, said task object corresponding to an operating system task.

58. The medium of claim 57 wherein said task object has at least one of a specified execution rate and priority.

59. The medium of claim 58 wherein at least two events with different tasks are executing in a model and wherein said instructions further comprise:

one or more instructions for using event transition components to schedule said execution of executable time-based components associated with said at least two events, said event transition components separating said execution of said executable time-based components associated with said at least two events.

60. The medium of claim 51 wherein said operation is controlled by an order of execution indicated a branch priority block.

61. The medium of claim 51 wherein said operation is said execution of more than one executable time-based component groups, said executable time-based component groups being a user selected grouping of said executable time-based components, said order of execution of said executable time-based component groups specified by a user.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.